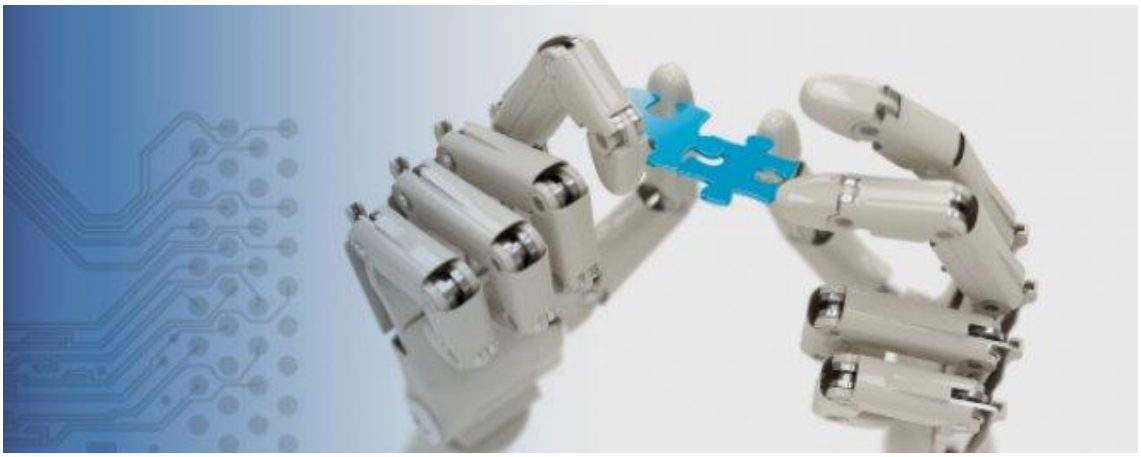
ROBOTICS MASTER

Universitat de Vic



Subject: Actuators and Controls

Session2: Low Level control

Exercise 2.1: PID Practice 1 Understand P, PI, PD and PID

Author: Toni Guasch Serra

Date: 2015-11-25

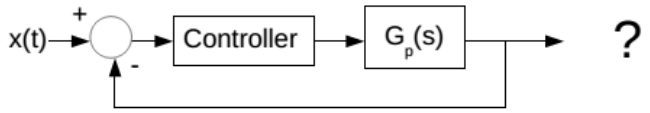
Exercise 2.1: PID Practice 1 Understand P, PI, PD and PID

Control:Controlling without a model

PID practice 1

Understand the effects of P, PI, PD and PID controllers on 1st and 2nd order plants

$x(t)=5*Unit\ Step$



$$G_p(s)=\frac{1}{s+1}$$

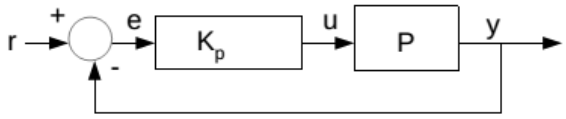
$$G_p(s)=\frac{1}{s^2+2s+12}$$

Provide system output and error signals obtained with different controllers and tunning parameters

P: Proportional Controller

Control: PID example

Pure proportional feedback: **The P controller**



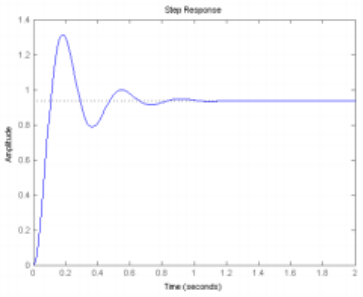
$$\frac{X(s)}{F(s)}=\frac{K_p}{s^2+10s+20K_p}$$

Closed-loop Steady state error using P controller

MATLAB

```
Kp = 300;
C = pid(Kp)
T = feedback(C*P,1)

t = 0:0.01:2;
step(T,t)
```



Reduced rise time

Reduced steady-state error

Increased overshoot

Decreased settling time by small amount

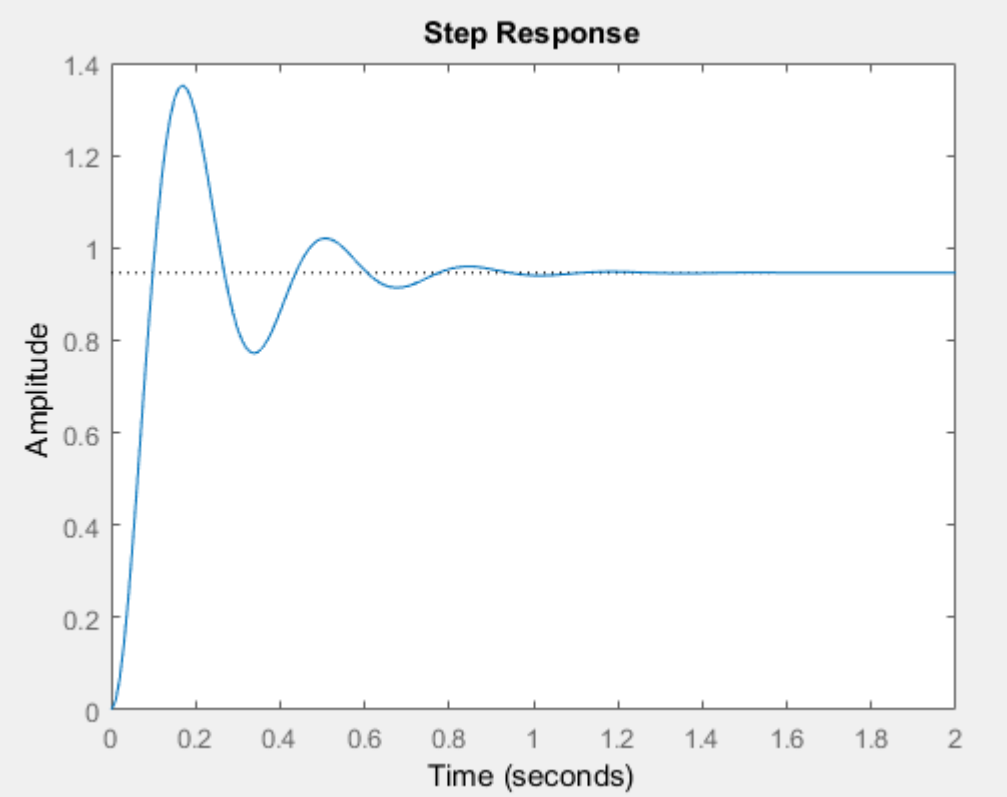
0) Matlab executed script:

Editor - C:\Users\Antoni Guasch\Documents\7 ESTUDIOS\Master Robótica\2 Modul2 Actuadors i Co

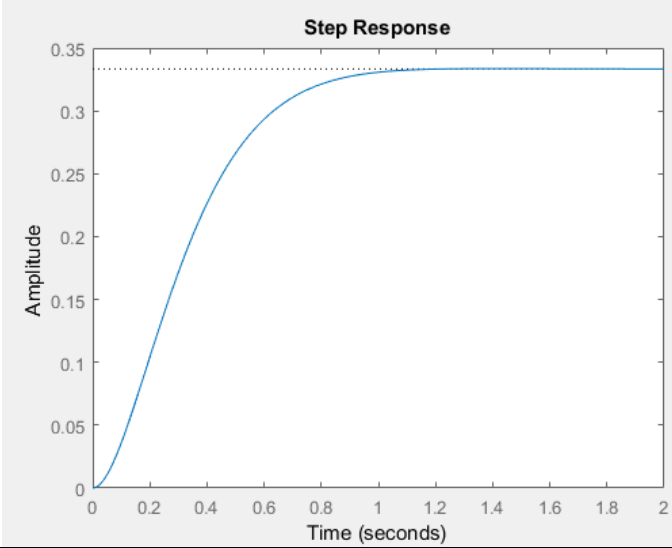
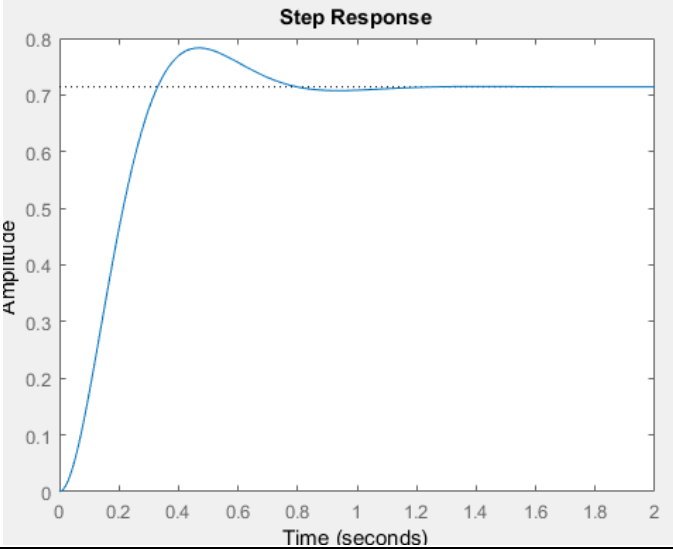
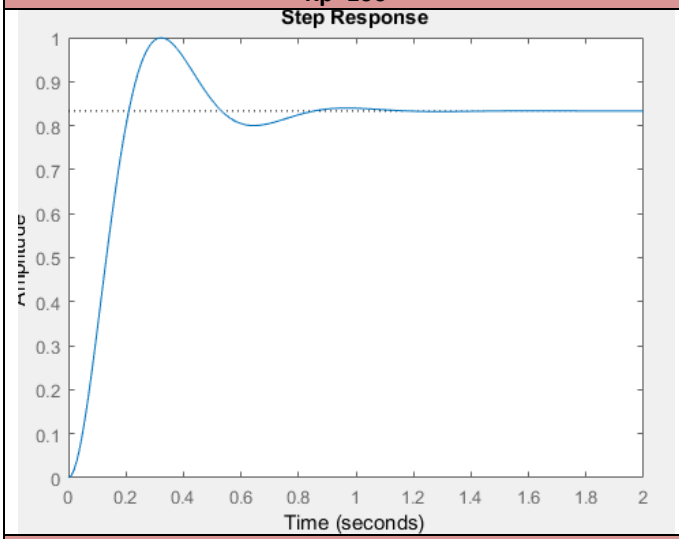
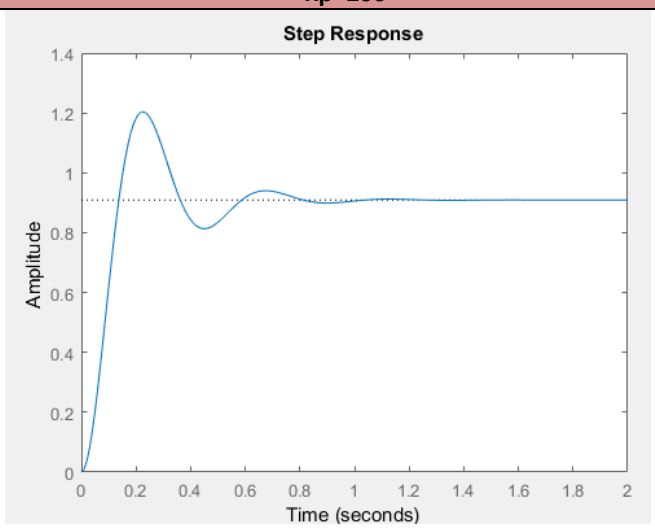
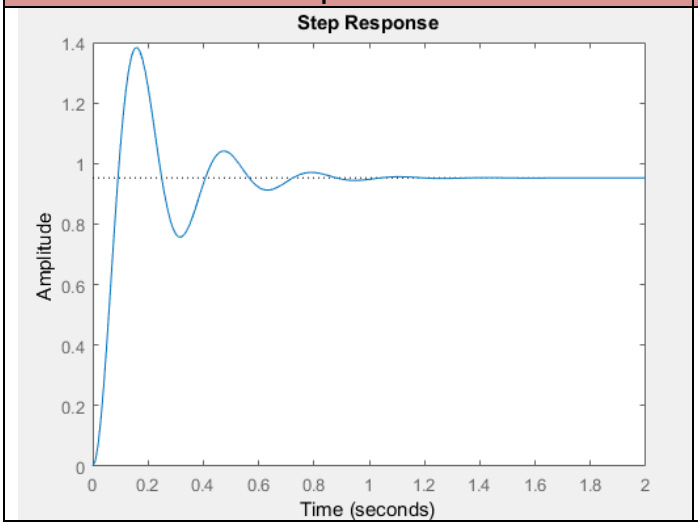
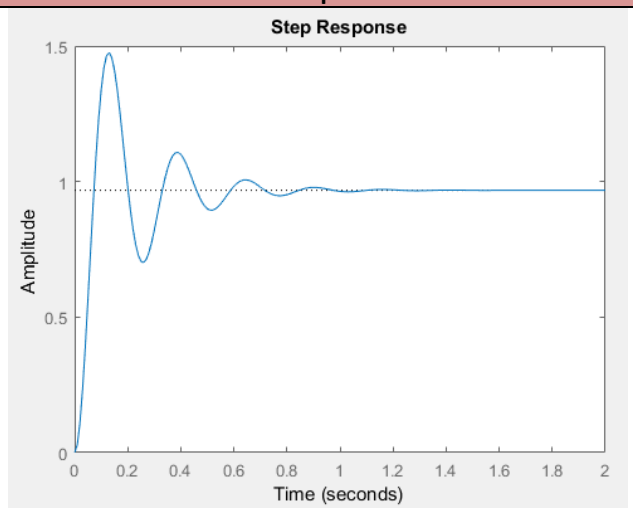
PIDControll_2.m | P_Controller.m | PD_Controller.m | PI_Controller.m

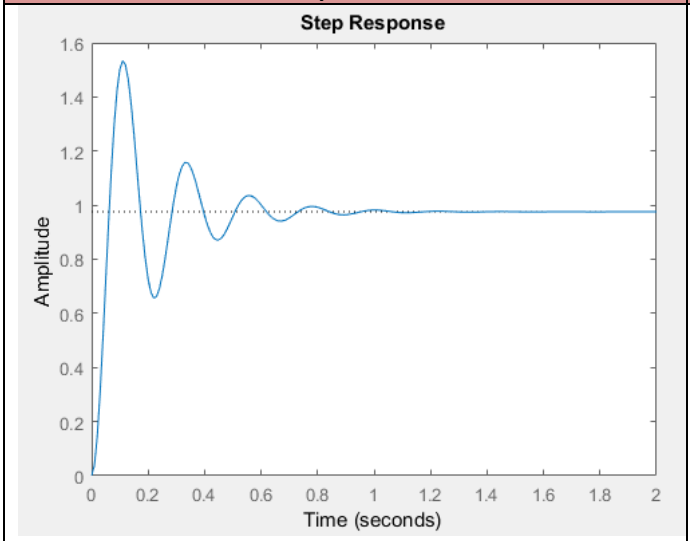
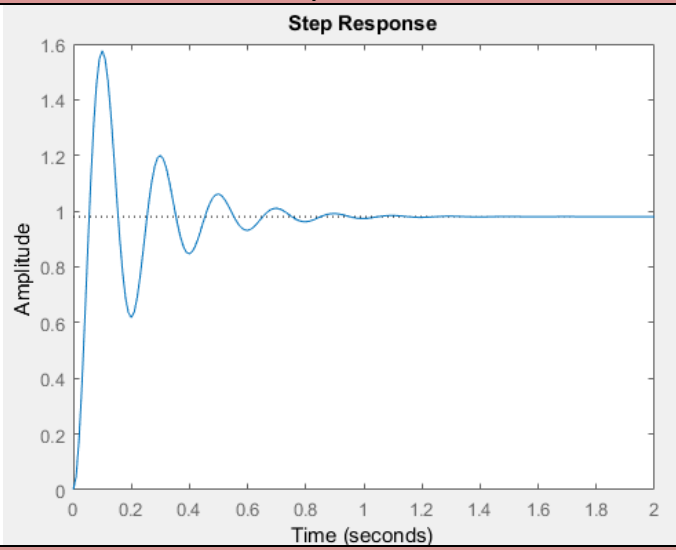
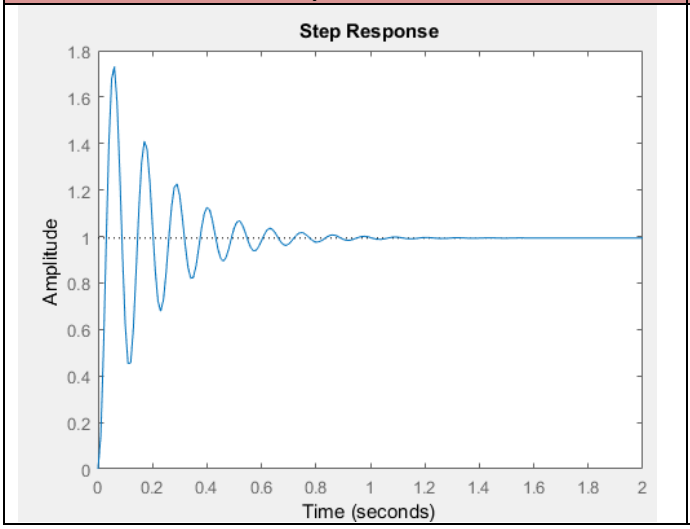
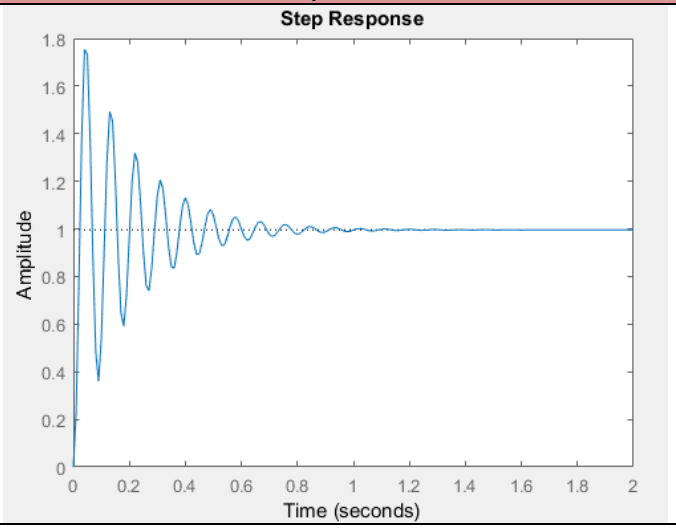
```
1 -    s = tf('s');
2 -    P = 1/(s^2 + 10*s + 20);
3      %step(P)
4
5 -    Kp = 300;
6 -    C = pid(Kp)
7 -    T = feedback(C*P,1)
8 -    t = 0:0.01:2;
9 -    step(T,t)
10
```

1) First test with the example values of the session 2 presentation:



2) Practice with other values of Kp

Kp=10	Kp=50
	
Kp=100	**Kp=200**
	
Kp=400	**Kp=600**
	

Kp=800	Kp=1000
	
Kp=3000	**Kp=5000**
	

The proportional term produces an output value that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a constant Kp, called the proportional gain constant.

Tuning the proportional controller gain (Kp) value we can see that if the value it is too low the response of the system is very slow under changes of the Setpoint, but stable under steady state conditions. And the amplitude is significantly lower than the unit. We have a less sensitive controller.

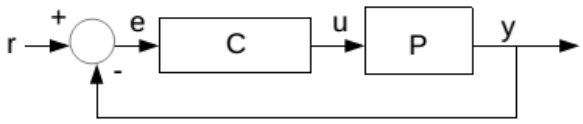
If we increase the gain of the proportional controller the response of the system is faster, but a value too high cause system instability with very aggressive oscillations.

The only way to get a quick response action to setpoint changes or "disruptions" in the process is to set an enough high constant gain until the appearance of an "overshoot" or on impulse.

PI

Control: PID example

Proportional - Integral feedback: **The PI controller**



$$u(t)=K_p e(t)+K_i\int e(\tau)d\tau$$

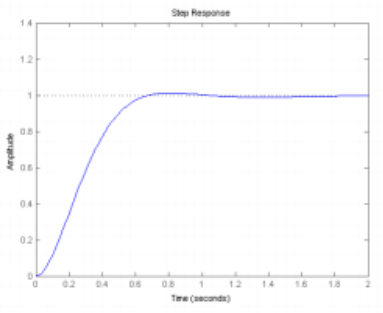
$$\frac{X(s)}{F(s)}=\frac{K_p s+K_i}{s^3+10s^2+K_p s+20+K_i}$$

Closed-loop Steady state error using PI controller

MATLAB

```
Kp = 30;
Ki = 70;
C = pid(Kp,Ki)
T = feedback(C*P,1)

t = 0:0.01:2;
step(T,t)
```



Eliminate steady-state error

As we decreased the P term, we have as well increased rise time a little

UVIC UNIVERSITAT DE VIC UNIVERSITAT CENTRAL DE CATALUNYA | eurecat | Master on Robotics Postgraduate in Industrial Robotics / Mobile Robotics | Josep M. Mirats Tur www.inlocrobotics.com | II.Actuators And Control | 27

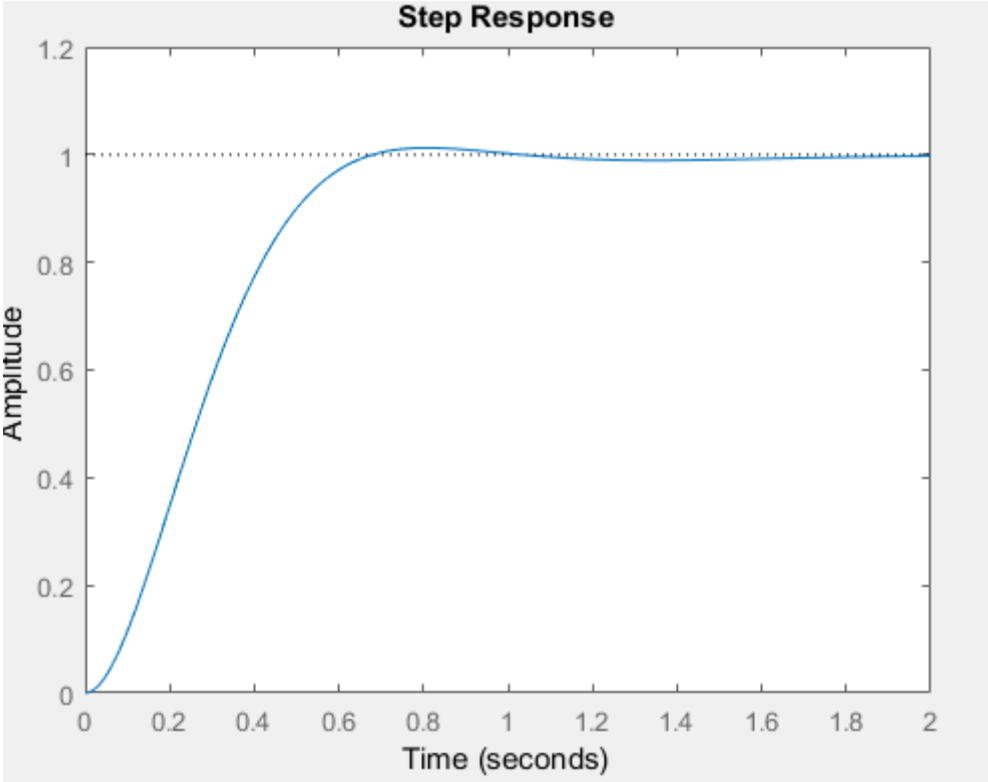
0) Matlab executed script:

Editor - C:\Users\Antoni Guasch\Documents\7 ESTUDIOS\Master Robótica\2 Modul2 Actuado

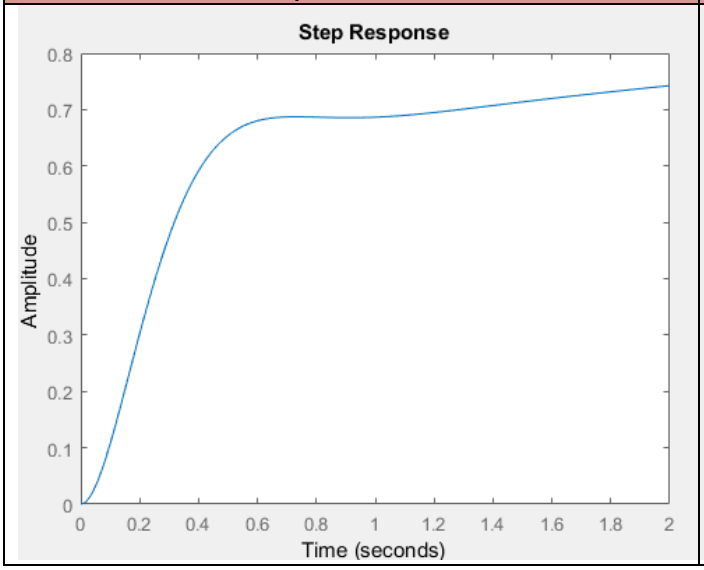
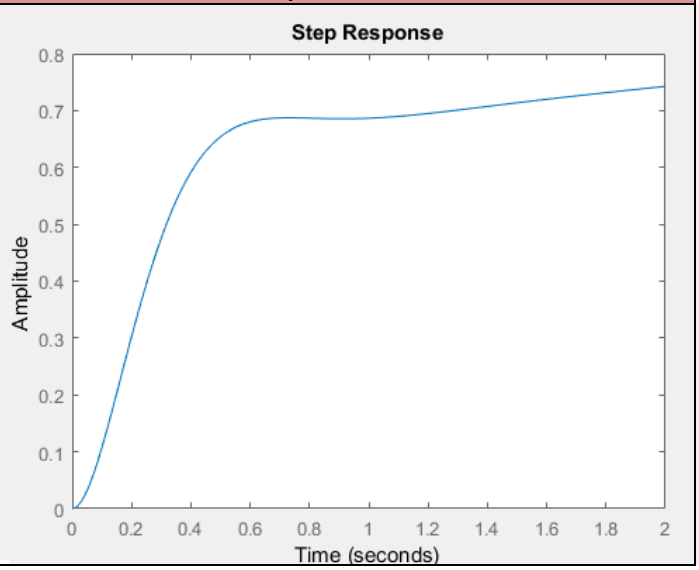
PIDControll_2.m | P_Controller.m | PD_Controller.m | PI_Controller.m*

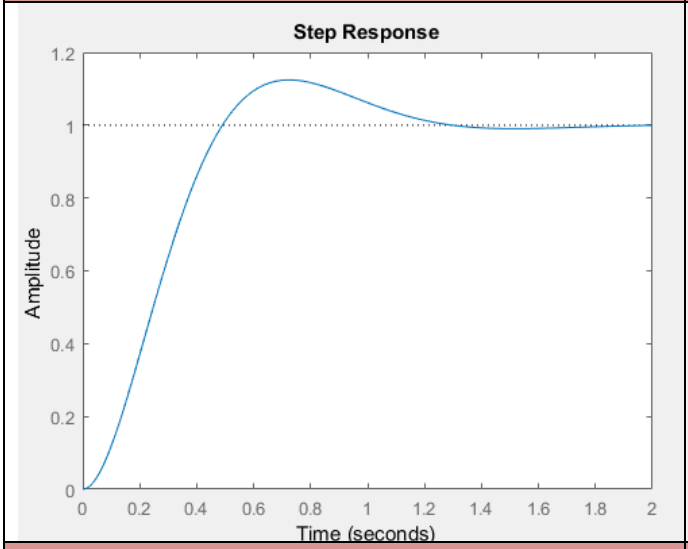
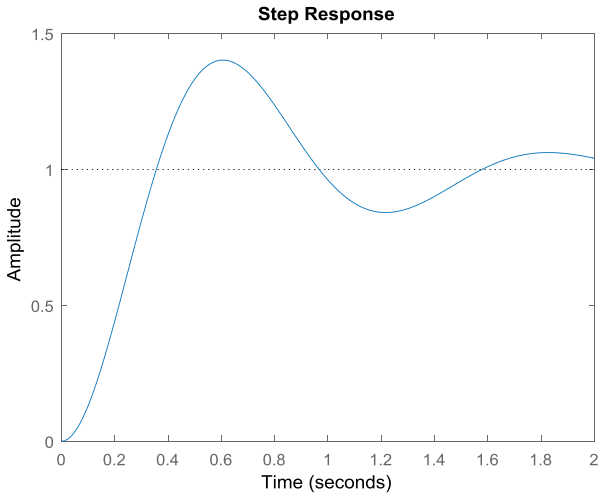
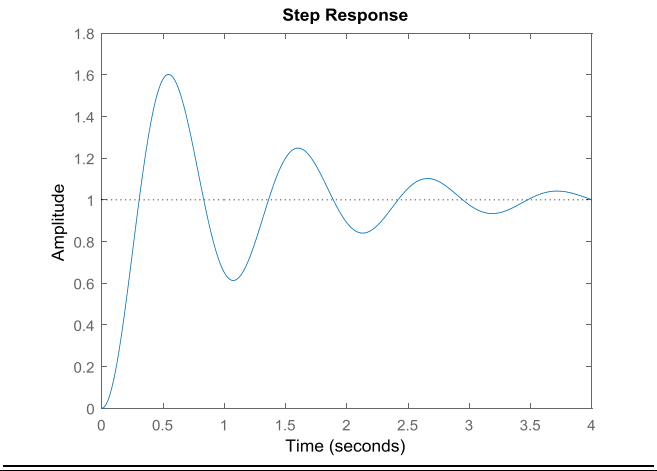
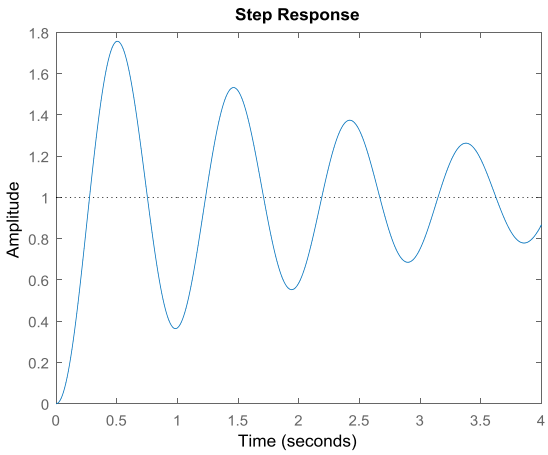
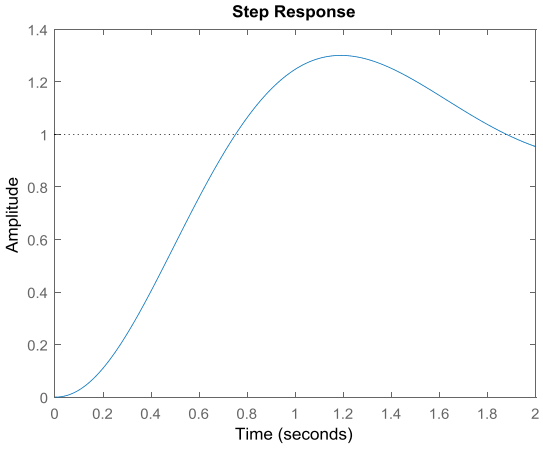
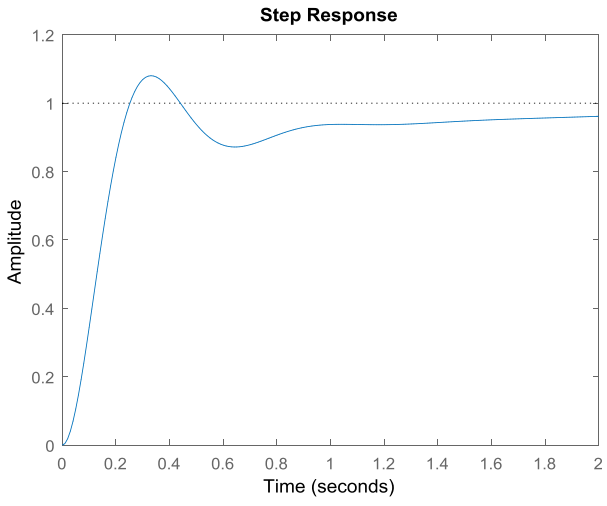
```
1
2 -     s = tf('s');
3 -     P = 1/(s^2 + 10*s + 20);
4 -     %step(P)
5
6 -     Kp = 30;
7 -     Ki = 70;
8 -     C = pid(Kp,Ki)
9 -     T = feedback(C*P,1)
10 -    t = 0:0.01:2;
11 -    step(T,t)
12
```

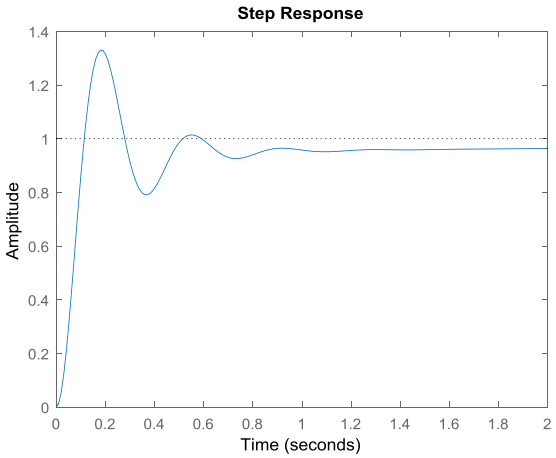
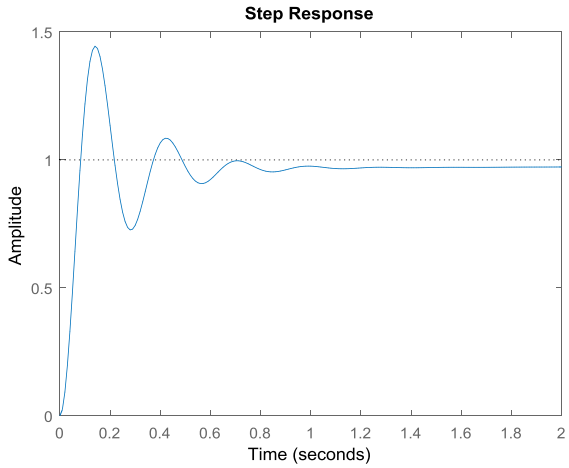
1) First test with the example values of the session 2 presentation:



2) Practice with other values of Kp and Ki

Kp=30, Ki=10	Kp=30, Ki=10
	

Kp=30, Ki=100	Kp=30, Ki=200
	
Kp=30, Ki=300	**Kp=30, Ki=400**
	
Kp=5, Ki=70	**Kp=100, Ki=70**
	

Kp=300, Ki=70	Kp=500, Ki=70
	

The contribution from the integral term is proportional to both the magnitude of the error and the duration of the error. The integral in a PID controller is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously. The accumulated error is then multiplied by the integral gain (Ki) and added to the controller output.

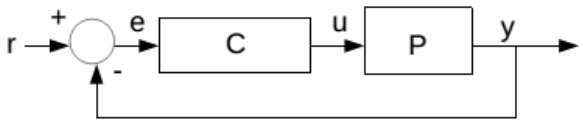
The integral term accelerates the movement of the process towards setpoint and eliminates the residual steady-state error that occurs with a pure proportional controller. However, since the integral term responds to accumulated errors from the past, it can cause the present value to overshoot the setpoint value

Tuning Ki values, we can see than increasing this value the rise time decrease but the overshoot and the settling time increase. With a correct Ki gain we can adjust the amplitude very close to the unit, with a value too high we increase again the overshoot and we have a big impact in the settling time

PD

Control: PID example

Proportional - derivative feedback: **The PD controller**



$$u(t)=K_p e(t)+K_d\frac{de(t)}{dt}$$

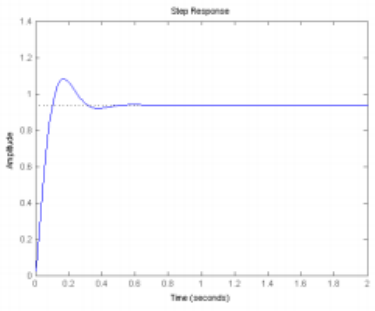
$$\frac{X(s)}{F(s)}=\frac{K_d s+K_p}{s^2+(10+K_d)s+(20+K_p)}$$

Closed-loop Steady state error using PD controller

MATLAB

```
Kp = 300;
Kd = 10;
C = pid(Kp,0,Kd)
T = feedback(C*P,1)

t = 0:0.01:2;
step(T,t)
```



Reduced overshoot

Reduced settling time

Small effect on rise time

Small effect on steady-state error

UVIC UNIVERSITAT DE VIC UNIVERSITAT CENTRAL DE CATALUNYA | eurecat | Master on Robotics Postgraduate in Industrial Robotics / Mobile Robotics | Josep M. Mirats Tur www.inlocrobotics.com | II.Actuators And Control | 26

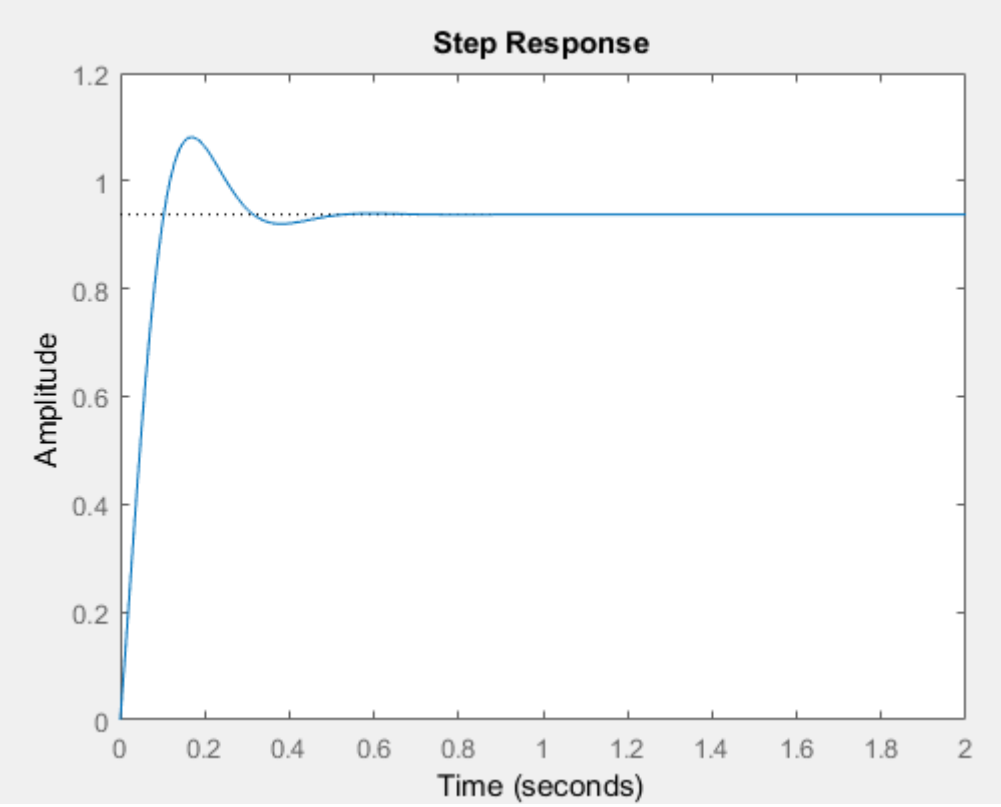
0) Matlab executed script:

Editor - C:\Users\Antoni Guasch\Documents\7 ESTUDIOS\Master Robótica\2 Modul2 Actuad

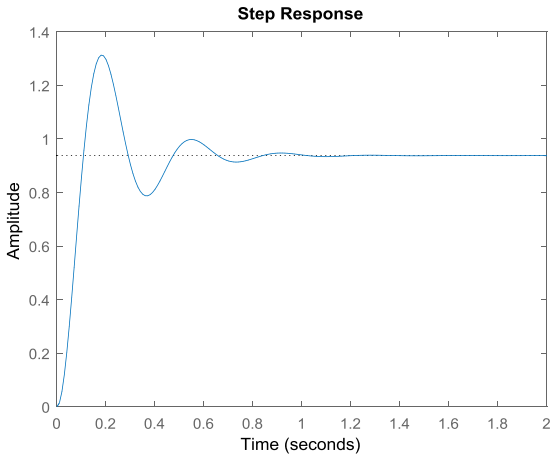
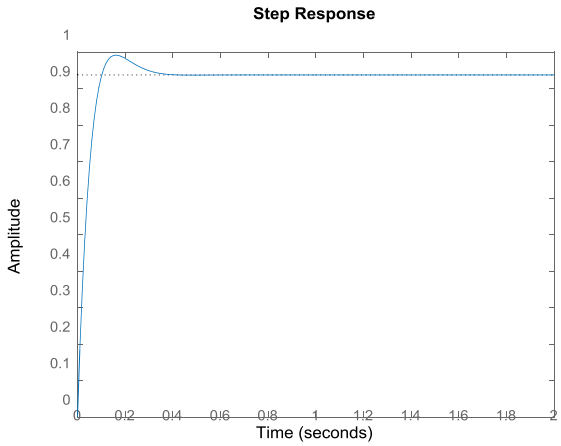
PIDControll_2.m | P_Controller.m | PD_Controller.m | PI_Controller.m

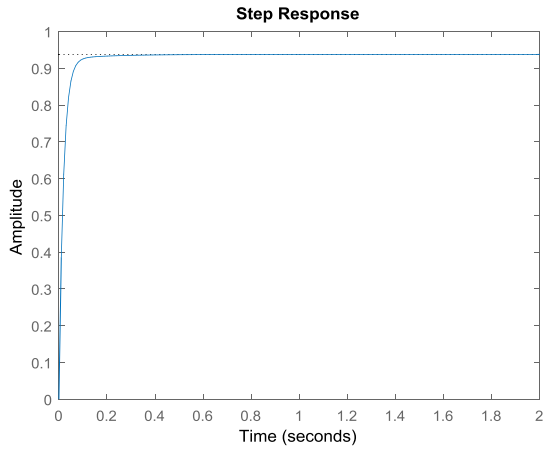
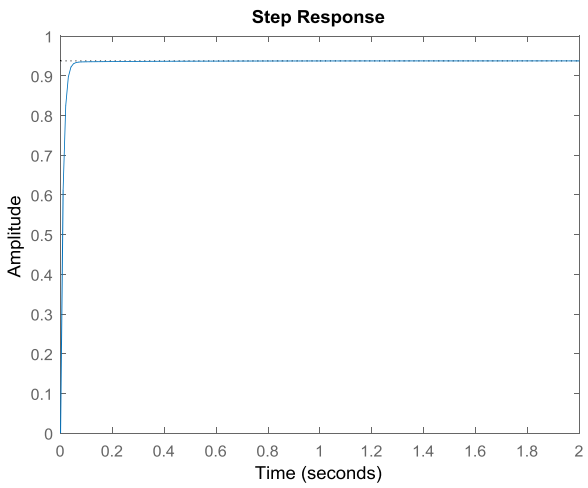
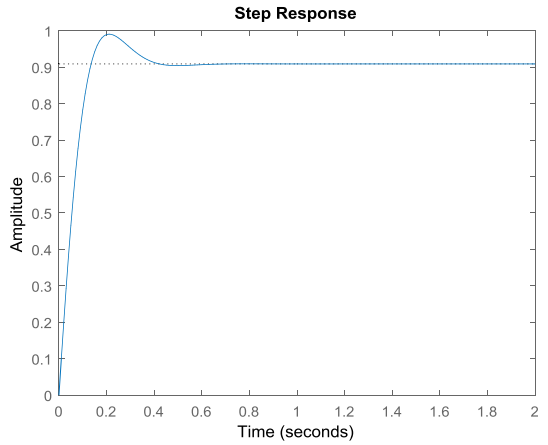
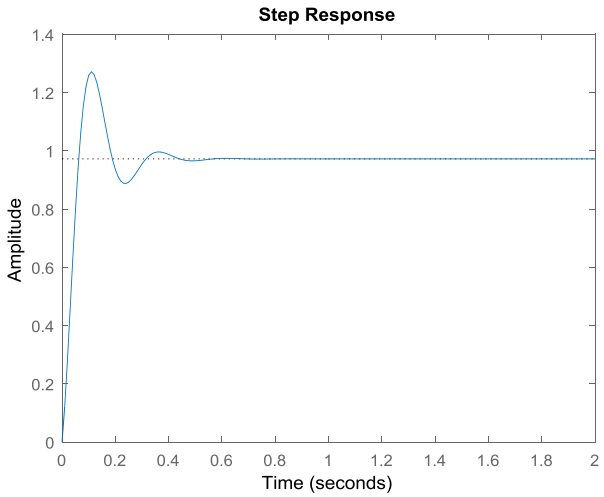
```
1
2 -     s = tf('s');
3 -     P = 1/(s^2 + 10*s + 20);
4       %Tstep(P)
5
6 -     Kp = 300;
7 -     Kd = 10;
8 -     C = pid(Kp,0,Kd)
9 -     T = feedback(C*P,1)
10 -    t = 0:0.01:2;
11 -    step(T,t)
```

1) First test with the example values of the session 2 presentation:



2) Practice with other values of Kp and Kd

Kp=300, Kd=0	Kp=300, Kd=20
	

Kp=300, Kd=50	Kp=300, Kd=100
	
Kp=200, Kd=10	**Kp=700, Kd=10**
	

The derivative of the process error is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain Kd. The magnitude of the contribution of the derivative term to the overall control action is termed the derivative gain, Kd.

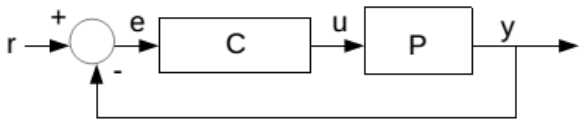
Derivative action predicts system behavior and thus improves settling time and stability of the system. An ideal derivative is not causal.

Tuning Kd values we have minor effect on the Rise time, but we have a very good improve in the Overshoot and Settling time decreasing it significantly.

PID

Control: PID example

The PID controller



$$u(t)=K_p e(t)+K_i\int e(\tau)d\tau+K_d\frac{de(t)}{dt}$$

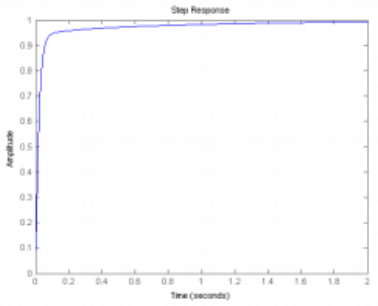
$$\frac{X(s)}{F(s)}=\frac{K_d s^2+K_p s+K_i}{s^3+(10+K_d)s^2+(20+K_p)s+K_i}$$

Closed-loop Steady state error using PID controller

MATLAB

```
Kp = 350;
Ki = 300;
Kd = 50;
C = pid(Kp,Ki,Kd)
T = feedback(C*P,1);

t = 0:0.01:2;
step(T,t)
```



No over-shoot

Fast rise time

No steady-state error

UVIC UNIVERSITAT DE VIC UNIVERSITAT CENTRAL DE CATALUNYA eurecat | Master on Robotics Postgraduate in Industrial Robotics / Mobile Robotics | Josep M. Mirats Tur www.inlocrobotics.com | II.Actuators And Control | 28

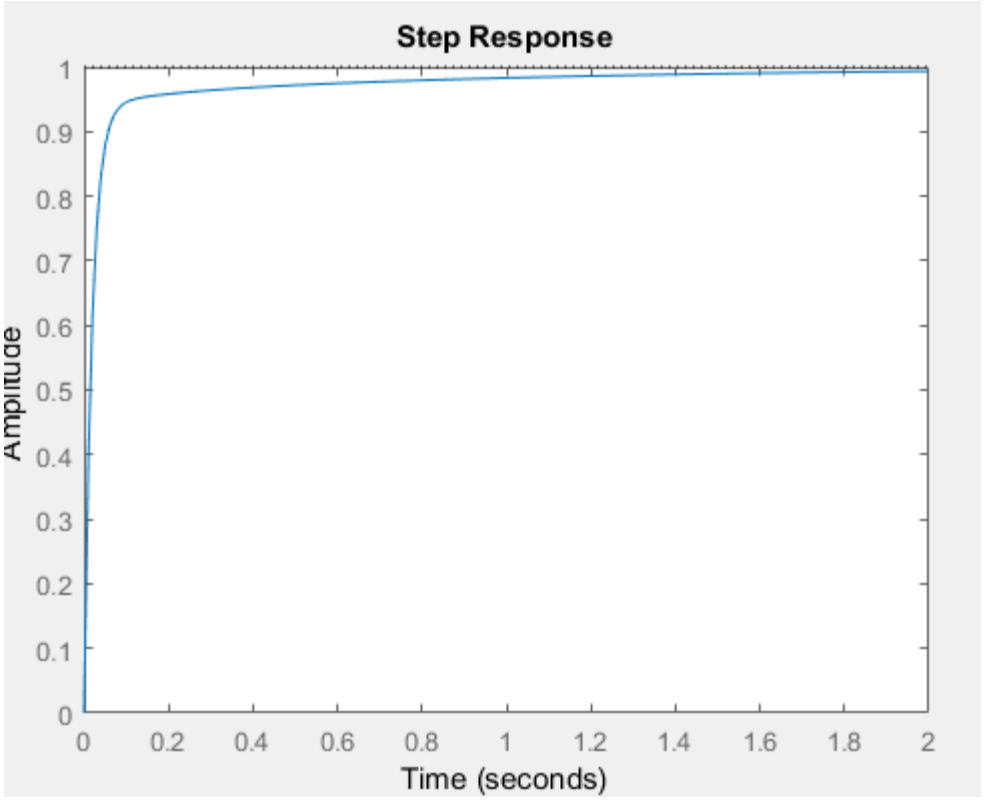
0) Matlab executed script:

Editor - C:\Users\Antoni Guasch\Documents\7 ESTUDIOS\Master Robótica\2 Modul2 Actuado

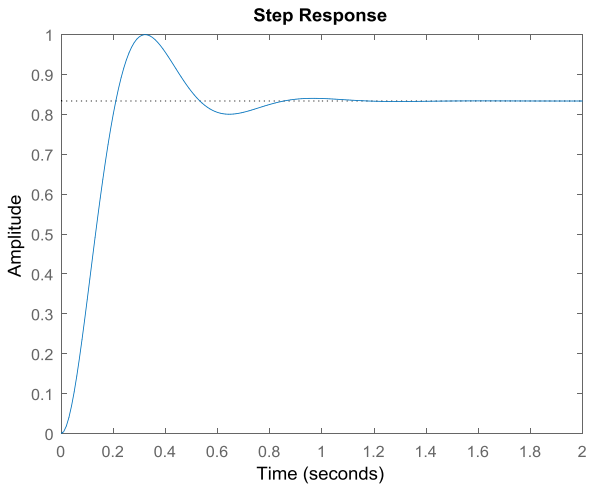
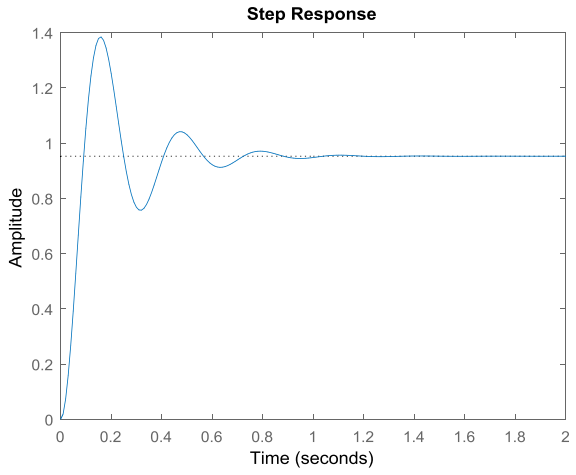
PIDControll_2.m | P_Controller.m | PD_Controller.m | PI_Controller.m

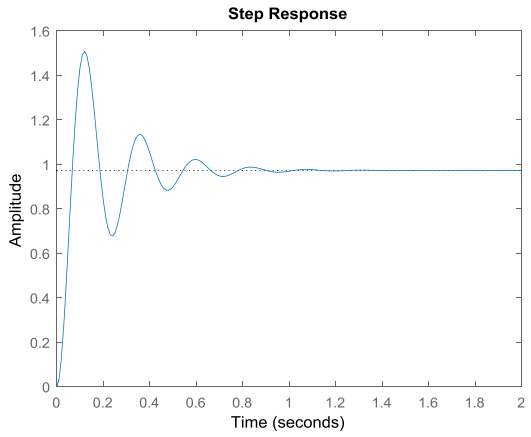
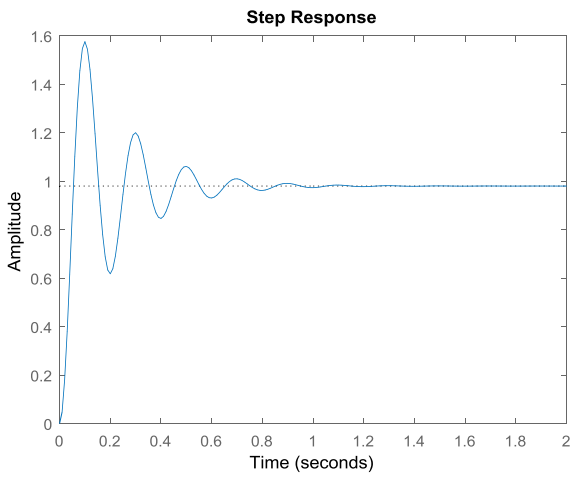
```
1 -  s = tf('s');
2 -  P = 1/(s^2 + 10*s + 20);
3    %step(P)
4
5 -  Kp = 350;
6 -  Ki = 300;
7 -  Kd = 50;
8 -  C = pid(Kp,Ki,Kd)
9 -  T = feedback(C*P,1);
10 -  t = 0:0.01:2;
11 -  step(T,t)
```

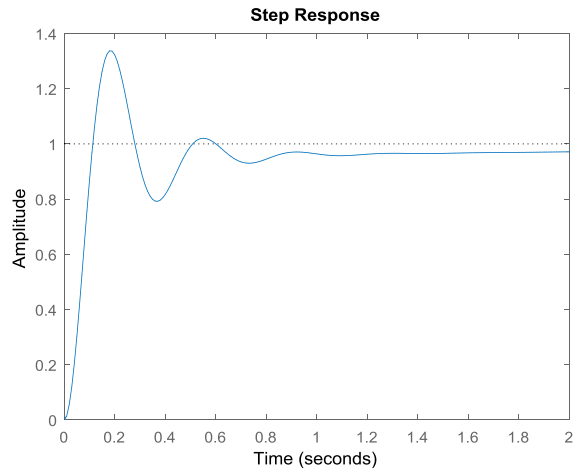
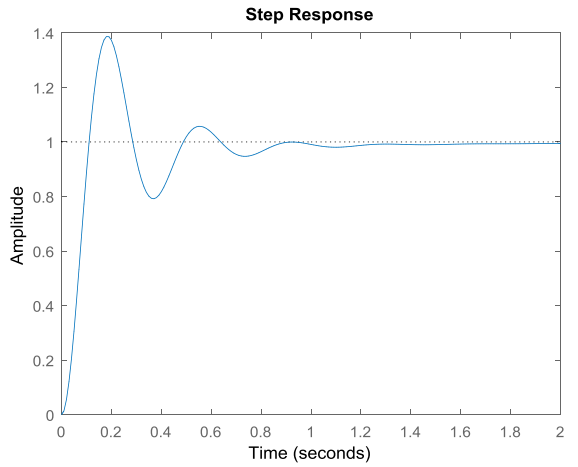
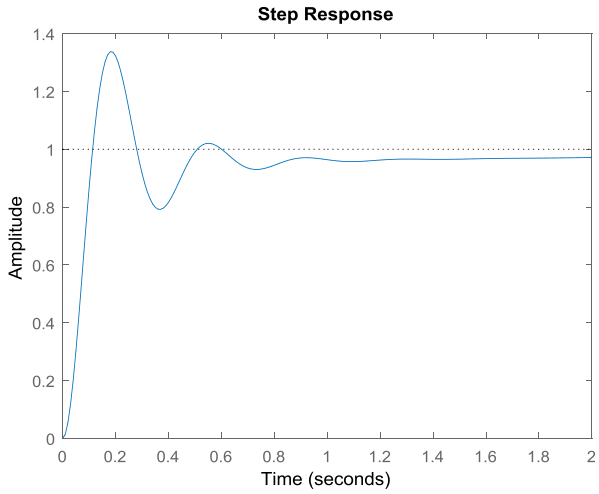
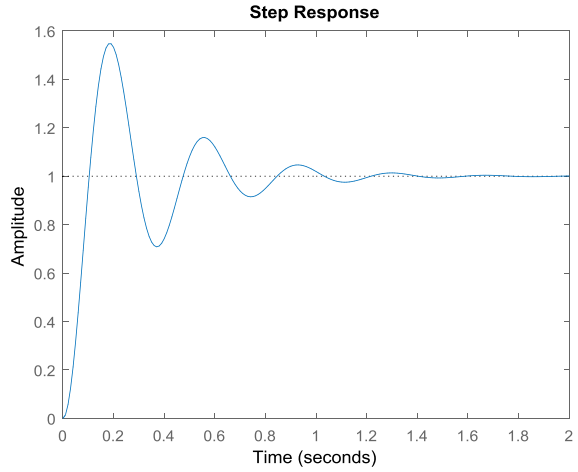
1) First test with the example values of the session 2 presentation:

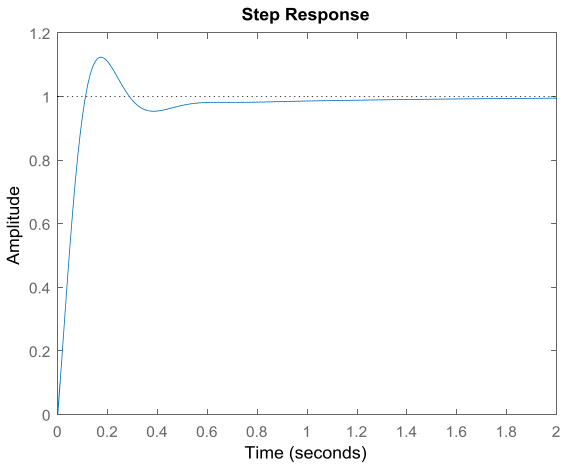
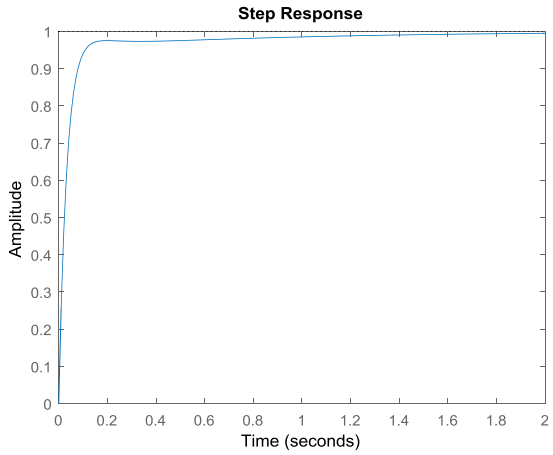
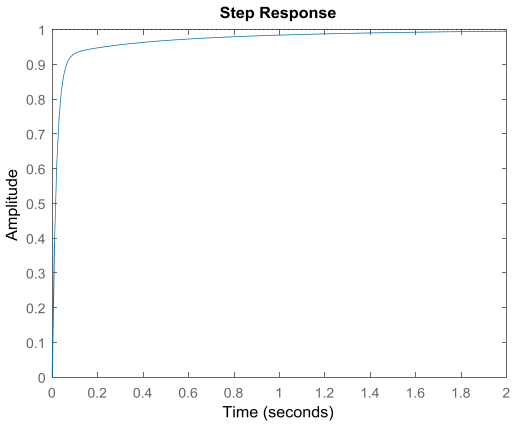
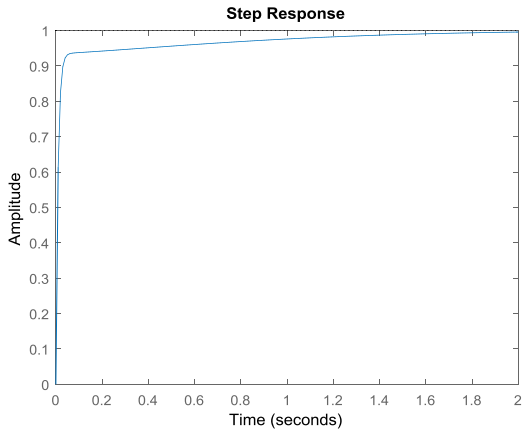


2) Practice with other values of Kp and Ki

1st: Search the optimum value of Kp	
Kp=100, Kd=0, Ki=0	**Kp=400, Kd=0, Ki=0**
	

Kp=700, Kd=0, Ki=0	Kp=100, Kd=0, Ki=0
	

2nd: with Kp adjusted to a value without too much oscilation. The following step is adjust Ki	
Kp=300, Kd=0, Ki=100	**Kp=300, Kd=0, Ki=300**
	
Kp=500, Kd=0, Ki=100	**Kp=300, Kd=0, Ki=1000**
	

3nd: with Ki adjusted. The last step is adjust Kp	
Kp=300, Kd=10, Ki=300	**Kp=300, Kd=30, Ki=300**
	
Kp=500, Kd=50, Ki=300	**Kp=300, Kd=100, Ki=30**
	

A PID controller mix the three controllers described until now

Due to this tuning of Kp, Ki and Kd we can see the effects of the Proportional, Derivative and Integral encoder working together. The tuning of these values is a key concept in the adjustment of the controller. If the PID controller parameters (the gains of the proportional, integral and derivative terms) are chosen incorrectly, the controlled process input can be unstable, i.e., its output diverges, with or without oscillation, and is limited only by saturation or mechanical breakage. Instability is caused by excess gain, particularly in the presence of significant lag.

- Proportional: for present values of the error (e.g. if the error is large and positive, the control variable will be large and negative)
- Integral: for past values of the error (e.g. if the output is not sufficient to reduce the size of the error, the control variable will accumulate over time, causing the controller to apply a stronger action), and
- Derivative: accounts for possible future values of the error, based on its current rate of change

For a manual tuning a good process to tune Kp, Ki and Kd is:

- First set Ki and Kd values to zero.

- Increase the Kp until the output of the loop oscillates, then the Kp should be set to approximately half of that value for a "quarter amplitude decay" type response.
- Then increase Ki until any offset is corrected in sufficient time for the process. However, too much Ki will cause instability.
- Finally, increase Kd, if required, until the loop is acceptably quick to reach its reference after a load disturbance. However, too much Kd will cause excessive response and overshoot.

A fast PID loop tuning usually overshoots slightly to reach the setpoint more quickly; however, some systems cannot accept overshoot, in which case an over-damped closed-loop system is required, which will require a Kp setting significantly less than half that of the Kp setting that was causing oscillation